EE4302
ADVANCED CONTROL SYSTEMS

Briefing Notes for CA1 and CA3: Computer-Aided Design of a State-Space Control System

CONTENTS

- Objective
- Simple State-Feedback Design *(CA1)*
  - Open-Loop System Analysis
  - Design Using Ackermann's Formula
  - Design Using Linear Quadratic Regulator (LQR)
- State-Feedback Design Introducing State-Augmentation *(CA3)*
- Key MATLAB Scripts
- Requirements and Questions for the Report
- Key References

OBJECTIVE

- Design a state-feedback controller for the second order plant specified by its state-space realization as follows.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1.10 & -3.01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- The Closed-Loop System needs to fulfill specified requirements on *bandwidth*, *resonant peak* and *steady state gain*. (See next page.)
- It is assumed that the state $x = [x_1 \quad x_2]^T$ is measurable at all time.

OBJECTIVE

- The specified requirements the designs are:

	Simple State-Feedback Design (CA1)	**State-Feedback Design Introducing State-Augmentation (CA3)**
Closed-Loop Bandwidth	≥ 3.5 dB	≥ 1.5 dB
Resonant Peak M_r	≤ 1.5 dB	≤ 2.0 dB
Steady State Gain Between r and y	0 dB	0 dB

SIMPLE STATE-FEEDBACK DESIGN

- OPEN-LOOP SYSTEM ANALYSIS

- Before the control system design, it is often good to look at the open-loop system performance and do a controllability check of the original plant.

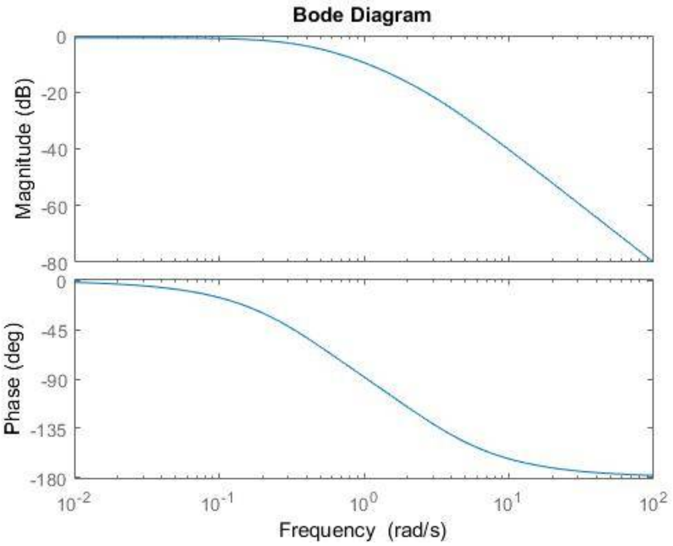
- The open-loop transfer function poles are located at: (calculate the eigenvalue of F)

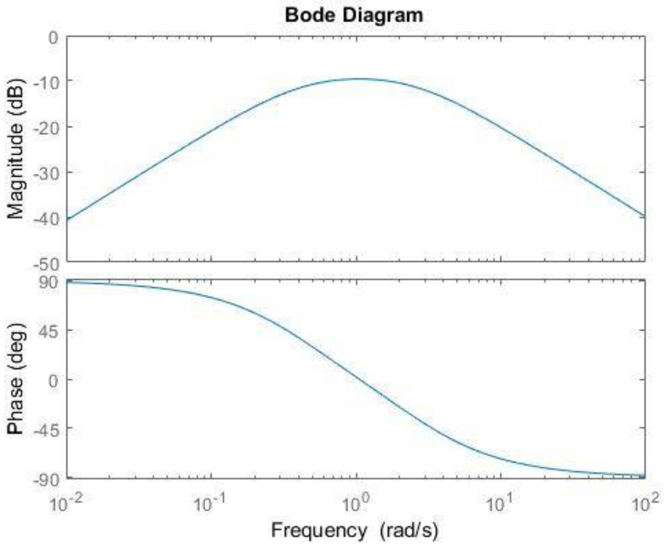
$$p_1 = -0.4256, p_2 = -2.5844$$

SIMPLE STATE-FEEDBACK DESIGN

- OPEN-LOOP SYSTEM ANALYSIS

- The frequency response plot for the original plant (from u to x_1 and x_2) are as follows. Left: x_1; Right: x_2.





SIMPLE STATE-FEEDBACK DESIGN

- OPEN-LOOP SYSTEM ANALYSIS

- The controllability matrix of the plant is:

$$\zeta = [G \quad FG] = \begin{bmatrix} 0 & 1 \\ 1 & -3.01 \end{bmatrix}$$

- The controllability matrix is non-singular, indicating that the original plant is controllable.

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- Consider a general SISO linear state-space model of a controllable plant in continuous time domain

$$\dot{x} = Fx + Gu$$
$$y = Hx \qquad 1$$

where $x \in \mathbb{R}^{n\times 1}$ is the state vector; $F \in \mathbb{R}^{n\times n}$ is the process matrix; $G \in \mathbb{R}^{n\times 1}$ and $F \in \mathbb{R}^{1\times n}$ are the input and output matrix respectively. u is the control signal applied to the plant and y is the output of the plant, both of which are scalars.

- Assume that the following control signal u is applied to the plant

$$u = -Kx + K_s r \qquad 2$$

where $K \in \mathbb{R}^{1\times n}$ is the controller gain; r is the reference signal; K_s is the scaling gain of r.

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- Substitute 2 into 1 to get the state-space model for the closed-loop system,

$$\begin{gathered}\dot{x} = Fx + G(-Kx + K_s r) = (F - GK)x + K_s r \\ y = Hx\end{gathered} \tag{3}$$

- From 3, we can see that by applying the control in 2, the process matrix of the closed-loop system becomes $(F - GK)$.

- Note that the transfer function for the closed-loop system in 3 is

$$H(s) = \frac{Y(s)}{R(s)} = H\big(sI - (F - GK)\big)^{-1} K_s \tag{4}$$

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

$$H(s) = \frac{Y(s)}{R(s)} = H\big(sI - (F - GK)\big)^{-1} K_s$$

- The poles of the transfer function are the eigenvalues of the process matrix, in this case the eigenvalues of $(F - GK)$.
- By appropriately designed K, the closed-loop system would have optimized pole positions, leading to desired system performance.

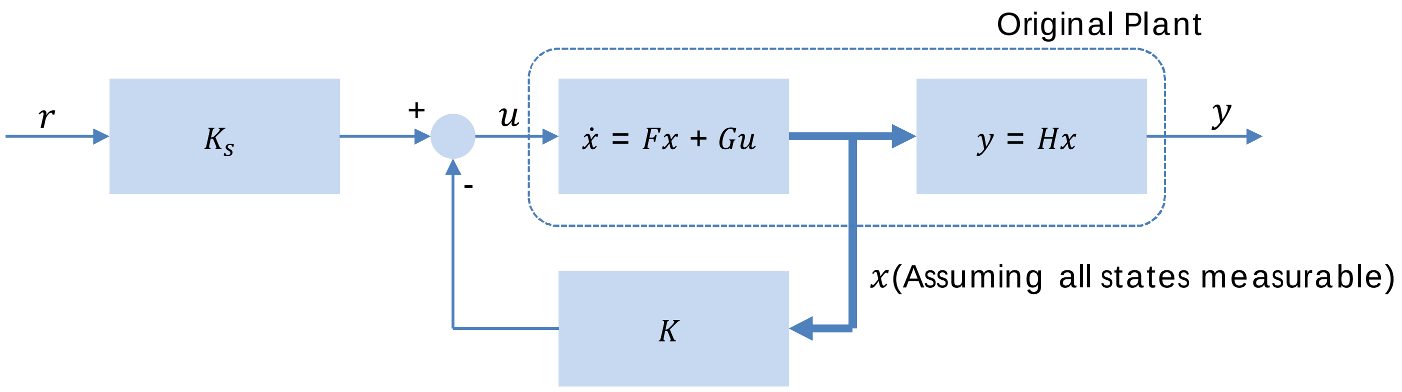
- K_s is a gain. After completing the design of K, K_s needs to be adjusted so that the steady state gain of the closed loop transfer function is 1 (i.e. no steady state error).

$$K_s = \frac{1}{\text{dcgain}\left(H\big(sI - (F - GK)\big)^{-1}\right)}$$

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- To get familiar with the control algorithm introduced by 2 , do the following simulation: arbitrarily choose $K = [0.1 \quad 0.1]$ and $K_s = 1$ in the controller and apply it to the original plant.
- The closed-loop system block diagram is shown as follows.

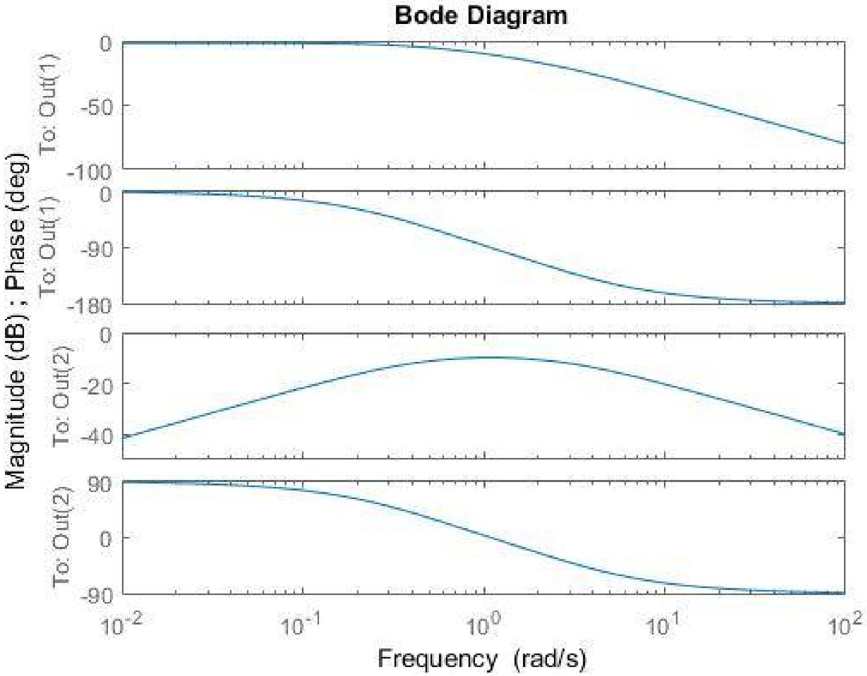


SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- The frequency response from r to x_1 and x_2 are shown in the following figure. It can also be calculated that the closed-loop transfer function poles from r to y becomes

$$p_1 = -0.4514, p_2 = -2.6586$$



SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- It has already been checked that (F, G) is controllable.

- It is possible do design K so that $(F - GK)$ has arbitrarily chosen eigenvalues, i.e. the closed-loop transfer function 4 , has any designed pole position.

- Ackermann’s formula is a general way of calculating such K.

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- Assume that the desired closed-loop pole polynomial is

$$\alpha_c(s) = s^n + \alpha_1 s^{n-1} + \cdots + \alpha_n \tag{6}$$

- The suitable controller gain K can then be calculated by

$$K = [0 \quad 0 \quad \cdots \quad 0 \quad 1]\zeta^{-1}\alpha_c(F) \tag{7}$$

where $\zeta = [G \quad FG \quad \cdots \quad F^{n-1}G]$ is the controllability matrix.

- Note that the controllability matrix must be non-singular, i.e. the plant must be controllable so that Ackermann’s formula is applicable.

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- Consider choosing the closed-loop poles based on ITAE table and Bessel table. The tables can be found in the Appendix of the briefing note.
- Note that there a bandwidth criterion of

  $$\text{Bandwidth} \geq 3.5\text{dB}$$

- Refer to prototype response poles table for ITAE and Bessel, the closed-loop poles are chosen as follows.

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = 4 \times \begin{bmatrix} -0.7071 + 0.7071i \\ -0.7071 - 0.7071i \end{bmatrix} \text{ (ITAE)}$$

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = 4 \times \begin{bmatrix} -0.8660 + 0.5000i \\ -0.8660 - 0.5000i \end{bmatrix} \text{ (Bessel)}$$

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- With determined pole position, the pole polynomial in 6 can be calculated and the controller gain K can be obtained using 7 .

- After the design of K, K_s can be calculated with 5 .

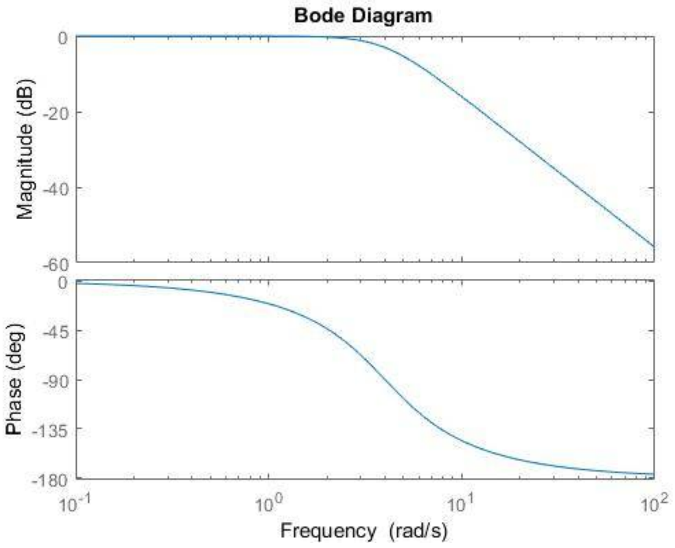
- The feedback controller gain K and the scaling gain K_s for ITAE and Bessel are concluded in the following table.

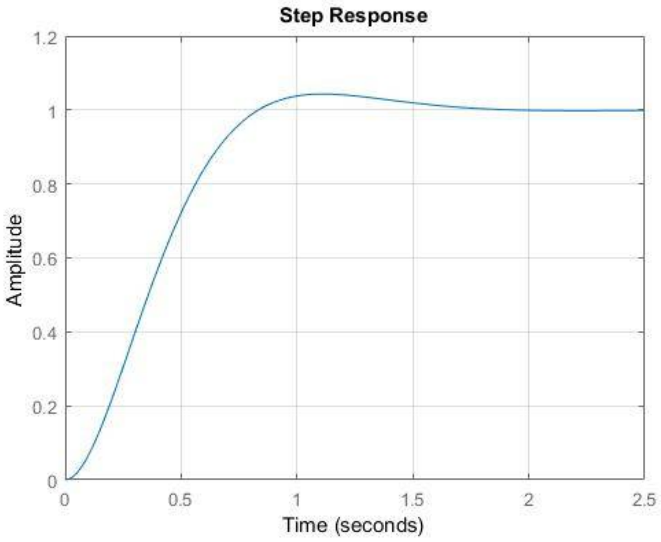
	ITAE	Bessel
K	[14.8997 2.6468]	[14.8993 3.9180]
K_s	15.9997	15.9993

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN’S FORMULA

- The frequency response and the unit step response for the designed closed-loop system with ITAE prototype pole position are as follows.

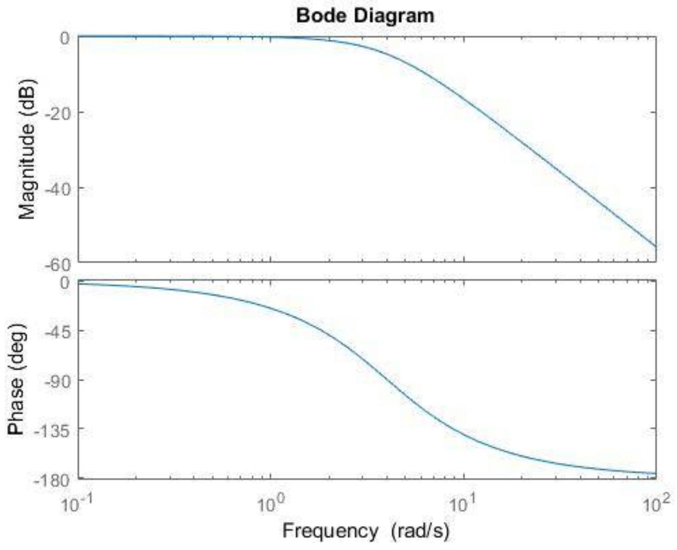


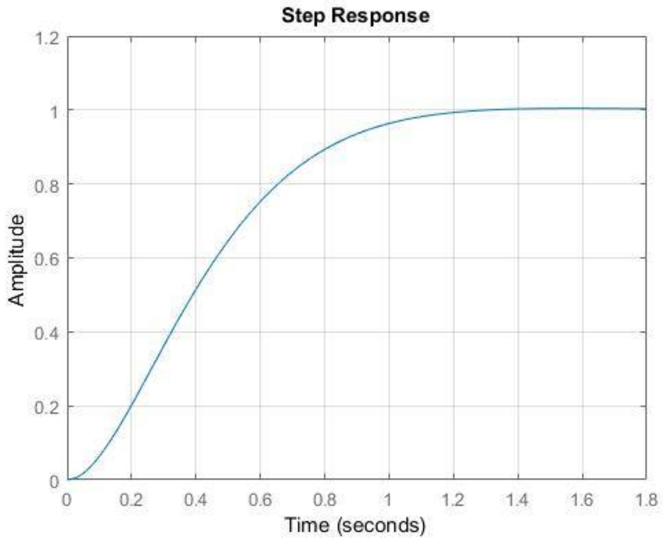


SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- The frequency response and the unit step response for the designed closed-loop system with Bessel prototype pole position are as follows.





SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- In the second-order dominant poles methodology, first of all we need to design a second-order reference model for the closed-loop system.
- A second-order transfer function with unity DC gain can be written in the following form.

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

- The detailed frequency domain analysis of the above transfer function can be found on classic textbooks. In this briefing note, only the part directly useful for our design is mentioned (see next page).

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

$$H(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

- We are considering under-damped case, i.e. $0 < \zeta < 1$
- Bandwidth:

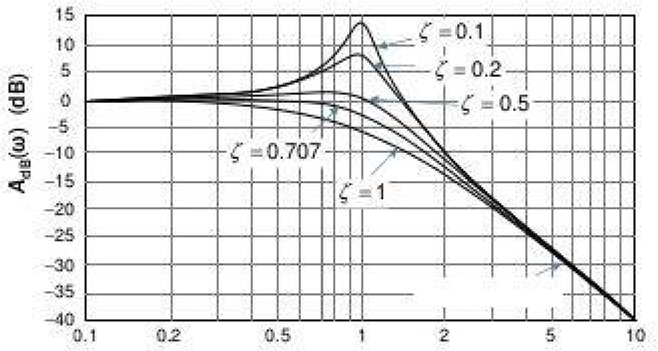
$$\omega_b = \omega_n \left((1 - 2\zeta^2) + \sqrt{(1 - 2\zeta^2)^2 + 1} \right)^{\frac{1}{2}}$$

- Resonant Peak:
  - $0 < \zeta \leq \frac{\sqrt{2}}{2}$

$$\omega_r = \omega_n \sqrt{1 - 2\zeta^2}, M_r = \frac{1}{2\zeta\sqrt{1 - \zeta^2}}$$

  - $\sqrt{2}/2 < \zeta < 1$

$A(\omega)$ monotonically decrease when $\omega > 0$



- Steady State Gain: Always 1

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- In the design, choose $\zeta = 0.8, \omega_n = 4.5$. The pole polynomial and the pole position of the reference model becomes

$$s^2 + 7.2s + 20.25 = 0$$

$$p_1 = -3.6000 + 2.7000i, p_2 = -3.6000 - 2.7000i$$

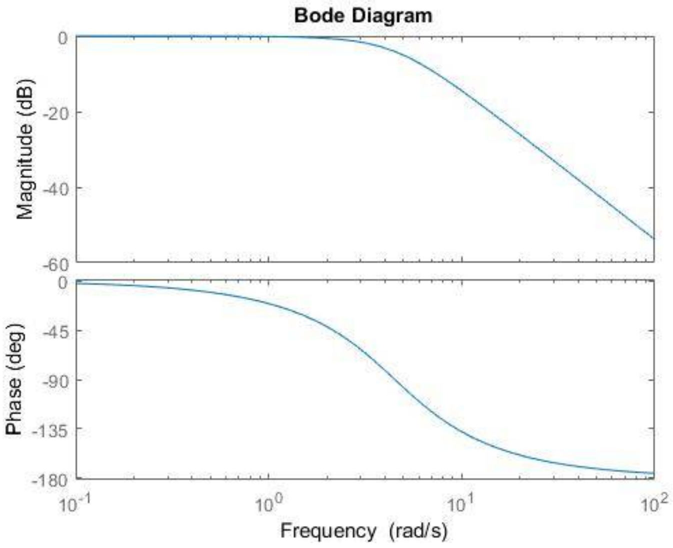
- Similarly with ITAE and Bessel prototype pole placement, the K and K_s can be calculated. The result is

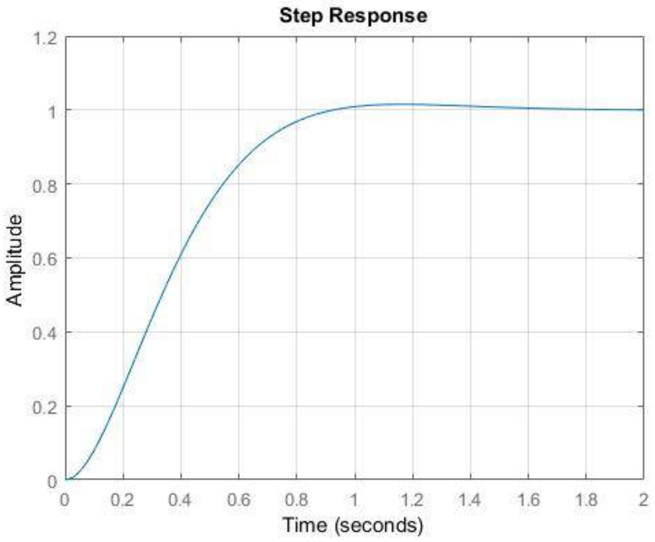
$$K = [19.1500 \quad 4.1900], K_s = 20.2500$$

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING ACKERMANN'S FORMULA

- The frequency response and the unit step response of the system is as follows.





SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING LINEAR QUADRATIC REGULATOR (LQR)

- In our design the LQR approach minimizes the following cost function (or performance index) during the regulation.

$$J = \int_0^\infty (x^T Q x + r u^2)$$

Note that $u^T R u$ is replaced by $r u^2$ since u is scalar in our example.

- The detailed mathematical proof, constrains and advantages of LQR can be found in classic textbooks, for instance, *(F.L. Lewis, Optimal Control)*.

- In our design, choose $Q = \text{diag}(q_1, q_2)$ with $q_1, q_2, r > 0$. Adjust these parameters to verify how the penalties setting would influence the closed-loop system performance.

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING LINEAR QUADRATIC REGULATOR (LQR)

- Choose

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, r = 1$$

- The feedback controller gain and the scaling gain can be calculated as

$$K = [0.3866 \quad 0.2814], K_s = 1.4866$$

- Some briefs in calculating K for LQR (not required by the report):
  - The following Riccati Equation is solved for symmetric positive definite P

$$F^T P + PF - \frac{1}{r} PGG^T P = -Q$$

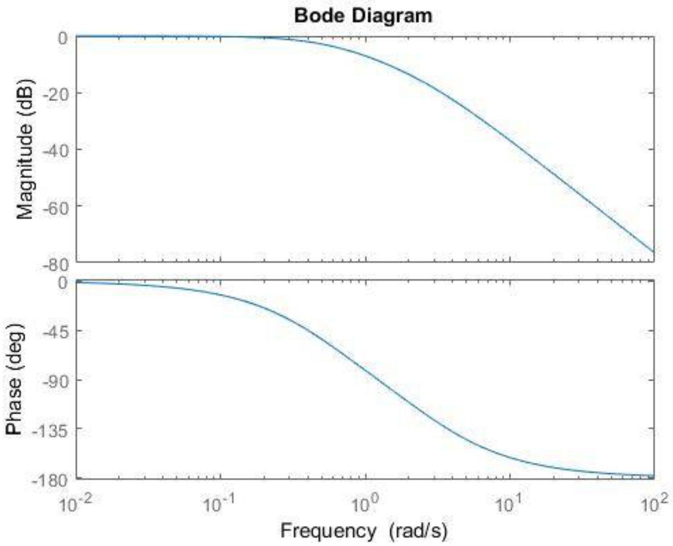
  - The feedback controller gain is then calculated by

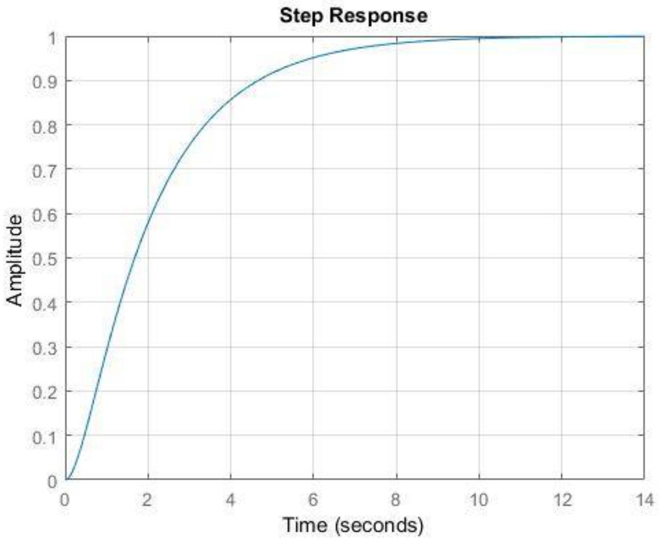
$$K = \frac{1}{r} G^T P$$

SIMPLE STATE-FEEDBACK DESIGN

- DESIGN USING LINEAR QUADRATIC REGULATOR (LQR)

- The frequency response and unit step response for the closed-loop system designed using LQR are as follows.





STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- Define a state variable x_I such that

$$\dot{x}_I = y - r = Hx - r$$

Note that x_I is not part of the original plant. It will be part of the closed-loop system.

- Define the joint state vector $\bar{x} = [x^T \quad x_I]^T$. The augmented state-space model of the system becomes

$$\dot{\bar{x}} = \begin{bmatrix} 0 & 1 & 0 \\ -1.10 & -3.01 & 0 \\ 1 & 0 & 0 \end{bmatrix} \bar{x} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} r \qquad 8$$

$$y = [1 \quad 0 \quad 0]\bar{x}$$

Let us name the matrix in the above equation $\bar{F}$, G_u, G_r and $\bar{H}$ respectively.

STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

$\bar{K} = [k_1 \quad k_2 \quad k_I]$

- If we can design $u = -\bar{K}\bar{x}$ for the system in 8 to stabilize the system, then at its steady state, there must be $\dot{\bar{x}} = 0$. Thus,

$$\dot{x}_I = y - r = 0, y = r$$

- Substitute $u = -\bar{K}\bar{x}$ into 8 . The closed-loop state-space model becomes

$$\begin{aligned} \dot{\bar{x}} &= (\bar{F} - G_u\bar{K})\bar{x} + G_r r \\ y &= \bar{H}\bar{x} \end{aligned} \qquad 9$$

- We can see from 9 that $\bar{F} - G_u\bar{K}$ becomes the process matrix of the closed-loop system.
- Similar with the “Simple Design”, we need to choose appropriate K so that $\bar{F} - G_u\bar{K}$ has desired eigenvalues, which would be the poles for the transfer function from r to y.

STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- There are plenty of ways to design $\bar{K}$. The solutions include ITAE and Bessel prototype pole placement, second-order dominant pole placement and LQR.

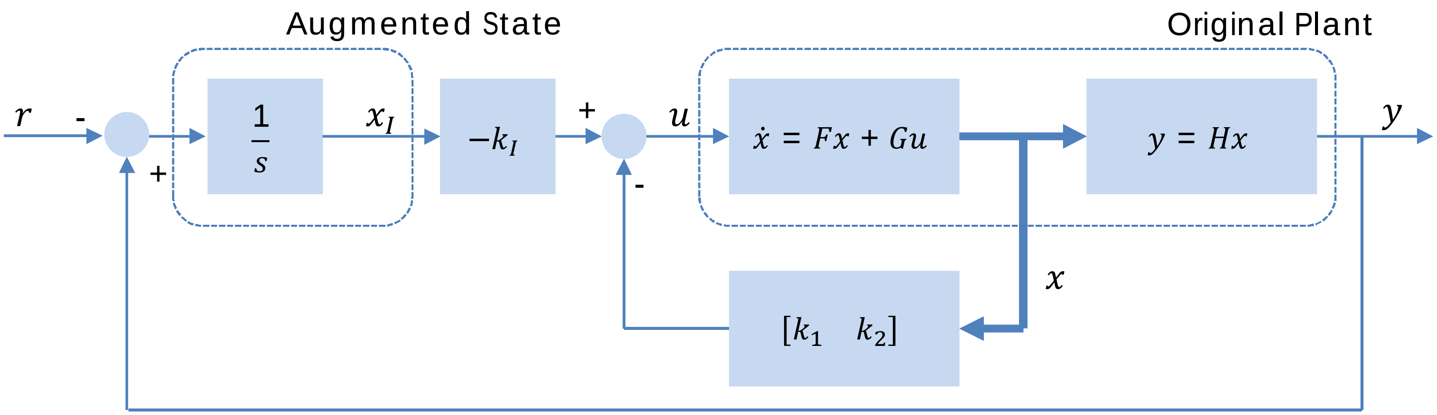
- We can apply the methodology in “Simple Design” to the augmented system. Note that this time the system is 3rd order.

- The control signal is

$$u = -\bar{K}\bar{x} = -k_1 x_1 - k_2 x_2 - k_I \int_0^t (y - r)\mathrm{d}t$$

Therefore, the closed-loop system block diagram is as follows (see next page).

STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION



STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- Consider using ITAE and Bessel prototype poles to do pole placement using Ackermann’s formula.

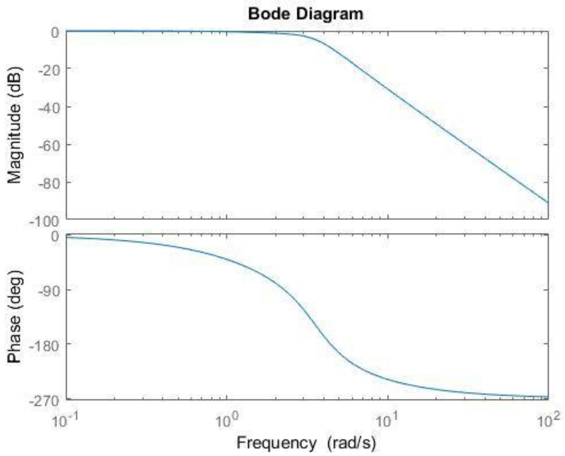
- The design and calculated controller gain are included in the following table. $\bar{K}$ is calculated using Ackermann’s formula.

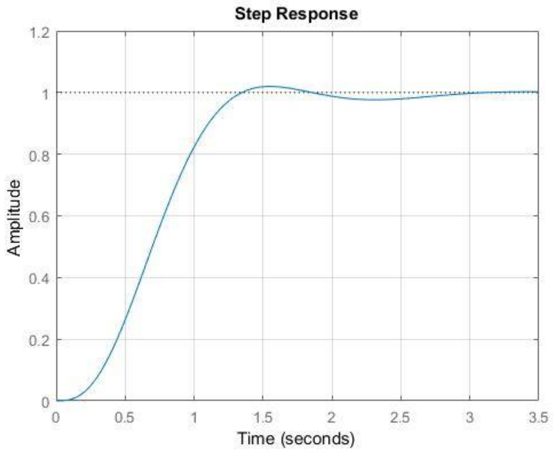
	ITAE	**Bessel**
Poles	$p_1 = -2.1243$ $p_{2,3} = -1.5630 \pm 3.2040i$	$p_1 = -2.8260$ $p_{2,3} = -2.2365 \pm 2.1336i$
$\bar{K}$	[18.2491 2.2403 26.9968]	[21.0949 4.2890 27.0001]

- The frequent domain response and unit step response is as follows (see next page).

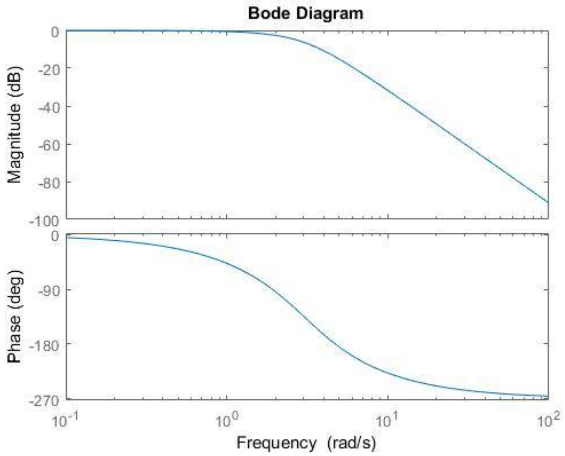
STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

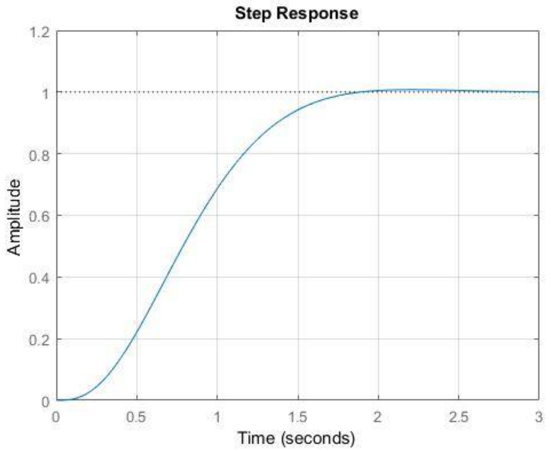
ITAE





Bessel





STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- In the design using second-order dominant pole approach, the same reference model is used. Therefore, the determined poles are

$$p_1 = -3.6000 + 2.7000i, p_2 = -3.6000 - 2.7000i$$

- Choose the third pole to be

$$p_3 = -12.000$$

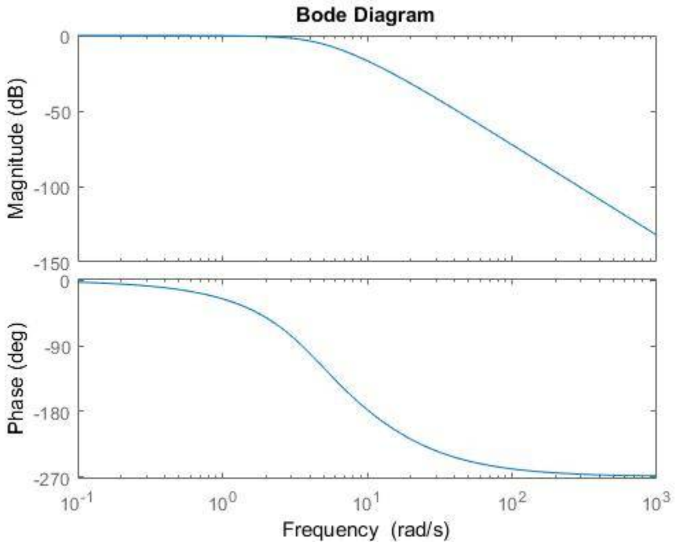
Note that the negative part of p_3 shall be much larger (in absolute value) than the dominant poles, 3 ~ 5 times be appropriate in practice.

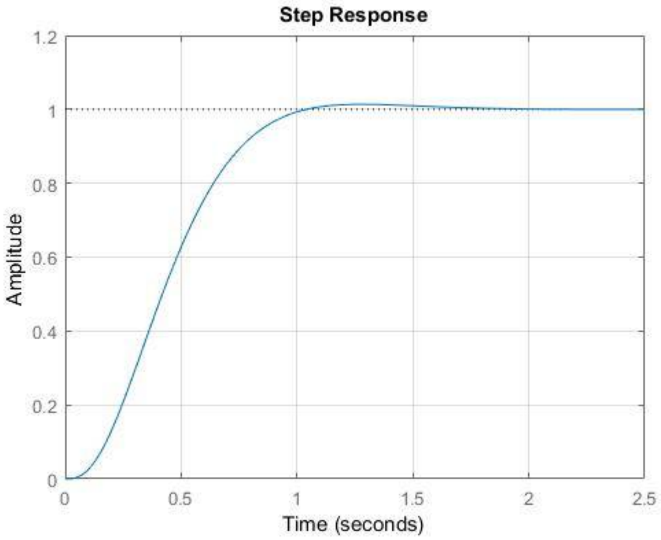
STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- The controller gain can be calculated using Ackermann’s formula.

$$\bar{K} = [105.55 \quad 16.19 \quad 243.00]$$

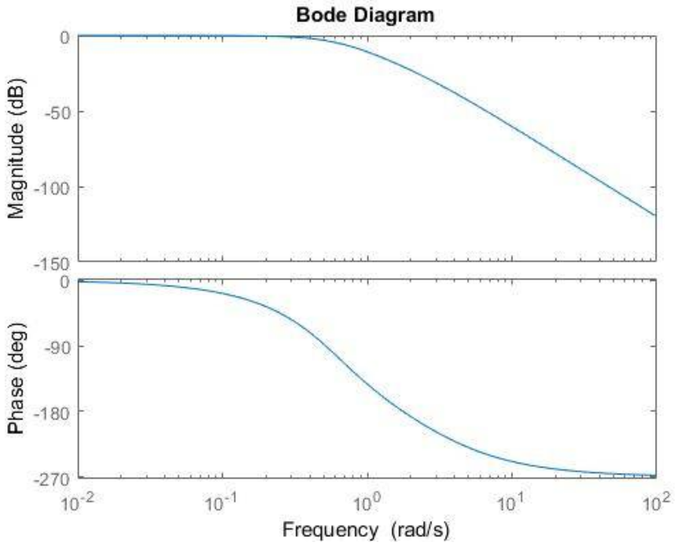
- The frequency response and unit step response of the closed-loop system are as follows.

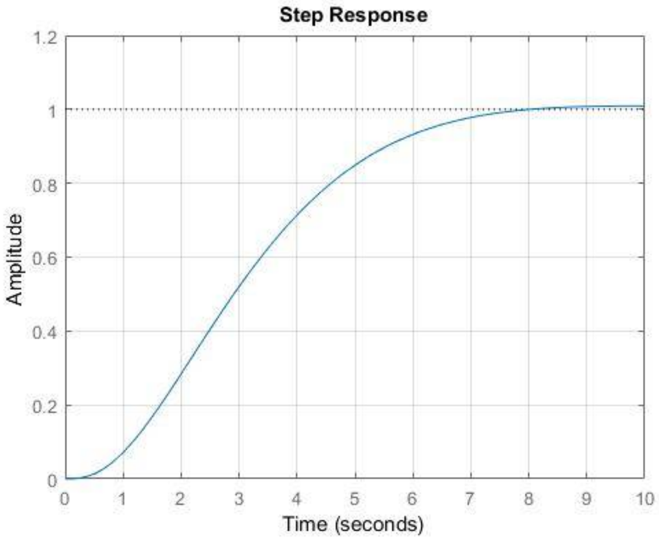




STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

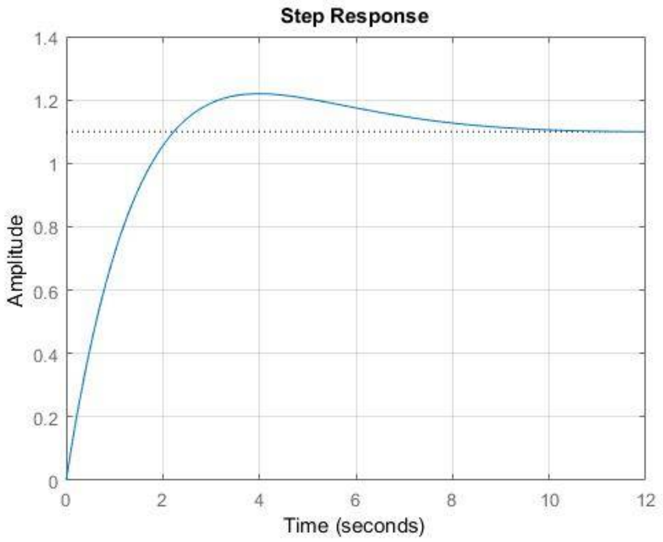
- In LQR design, choose $Q = I_{3\times3}$, $r = 1$. The controller gain then is
$$\overline{K} = [2.0175 \quad 0.7443 \quad 1.0000]$$
- The frequency response and unit step response of the closed-loop system are





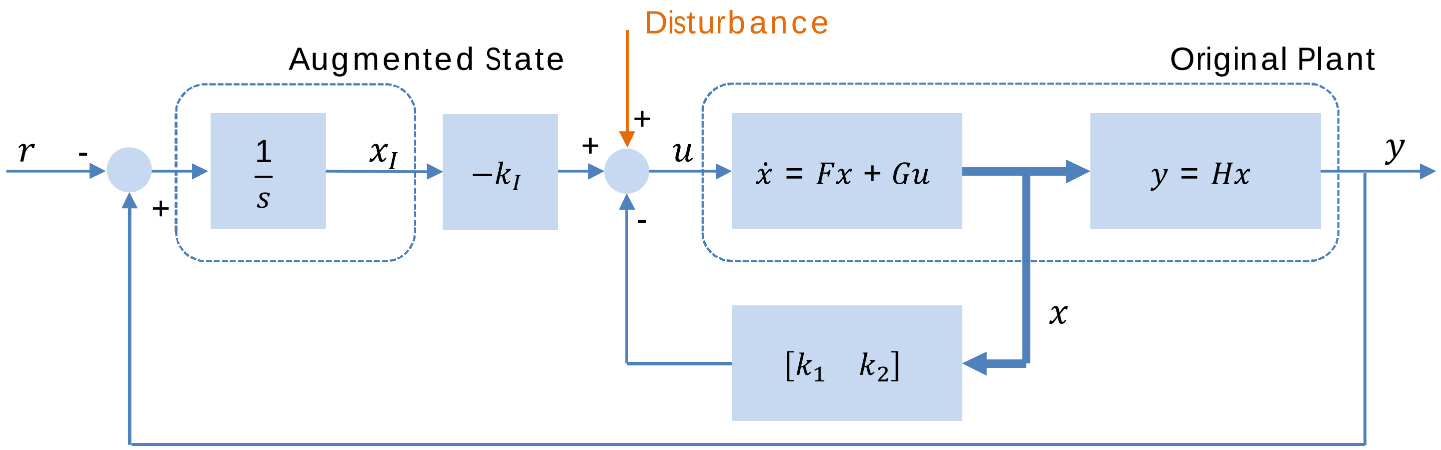
STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- The control signal for LQR with unit step reference signal is



STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- Consider an unmeasurable constant disturbance comes into the augmented-state system. The disturbance is mixed with the control signal u.



STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- With the disturbance, the augmented state-space model 8 becomes

$$\dot{\bar{x}} = \begin{bmatrix} 0 & 1 & 0 \\ -1.10 & -3.01 & 0 \\ 1 & 0 & 0 \end{bmatrix} \bar{x} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} r + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} v \qquad 9$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \bar{x}$$

Note that the disturbance is represented by v, which is assumed to be constant or low-frequent. The entrance matrix of v, G_v equals to G_u.

- What is the influence of the disturbance to the output? Would it cause a steady-state bias?

STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- It is obvious either from the block diagram or from 9 that the augmented system is linear to reference signal r and disturbance v.

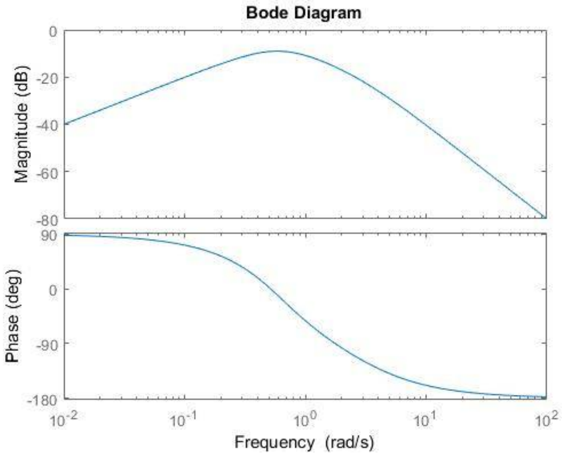
- Therefore, by superposition theory, the study of mixed simultaneous input r and v can be done separately.

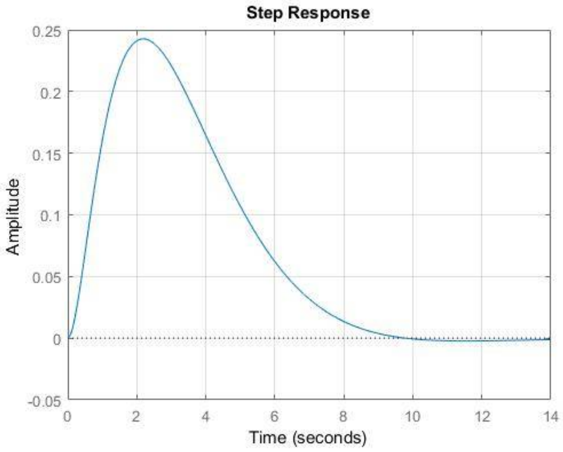
- We already studied the case where r is unit step with $v = 0$. The output y can track r with appropriate speed and no steady-state error.

- Now, we need to set $r = 0$ and v some value as the input to the closed-loop system. By observing y, we can see how the output is influenced by the disturbance.

STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

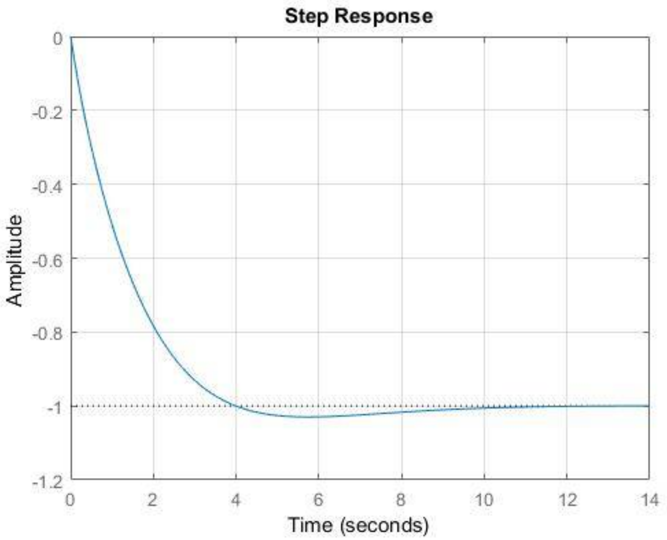
- The LQR design is used as an example. The control signal is $u = -\bar{K}\bar{x}$ where $\bar{K}$ is designed previously using LQR approach.
- The reference signal is $r = 0$.
- The frequency response and unit step response of y to v is shown below.





STATE-FEEDBACK DESIGN INTRODUCING STATE-AUGMENTATION

- The control signal, with the unit step disturbance, is shown below.



KEY MATLAB SCRIPTS

- The MATLAB program used in this briefing notes are included in the following few slides. **All the figures and calculation results in this briefing note can be obtained by running these MATLAB codes** (Comments in greed color is not necessary in executing the codes).

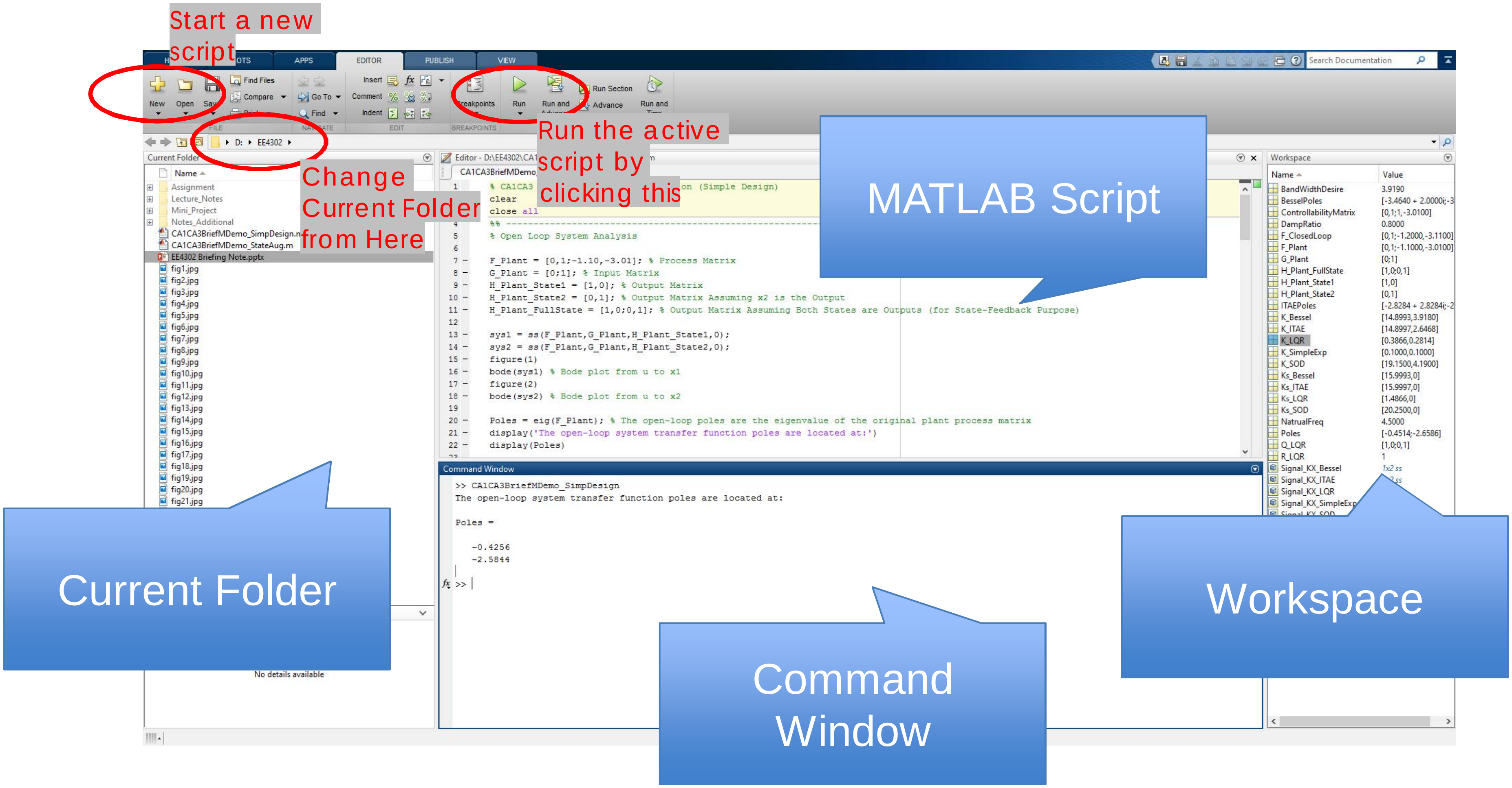
- The MATLAB program serves as an instruction to the use of MATLAB to carry out the simulations required by the module.

- The students need to add additional case studies by changing the value of some parameters in the program.

- The MATLAB program has been tested on MATLAB R2016b

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS



Start a new script

Change Current Folder from Here

Run the active script by clicking this

MATLAB Script

Current Folder

Command Window

Workspace

Current Folder:

- Assignment
- Lecture_Notes
- Mini_Project
- Notes_Additional
- CA1CA3BriefMDemo_SimpDesign.m
- CA1CA3BriefMDemo_StateAug.m
- EE4302 Briefing Note.pptx
- fig1.jpg
- fig2.jpg
- fig3.jpg
- fig4.jpg
- fig5.jpg
- fig6.jpg
- fig7.jpg
- fig8.jpg
- fig9.jpg
- fig10.jpg
- fig11.jpg
- fig12.jpg
- fig13.jpg
- fig14.jpg
- fig15.jpg
- fig16.jpg
- fig17.jpg
- fig18.jpg
- fig19.jpg
- fig20.jpg
- fig21.jpg

```
% CA1CA3                                        on (Simple Design)
clear
close all
%% ------------------------------------------------------------
% Open Loop System Analysis

F_Plant = [0,1;-1.10,-3.01]; % Process Matrix
G_Plant = [0;1]; % Input Matrix
H_Plant_State1 = [1,0]; % Output Matrix
H_Plant_State2 = [0,1]; % Output Matrix Assuming x2 is the Output
H_Plant_FullState = [1,0;0,1]; % Output Matrix Assuming Both States are Outputs (for State-Feedback Purpose)

sys1 = ss(F_Plant,G_Plant,H_Plant_State1,0);
sys2 = ss(F_Plant,G_Plant,H_Plant_State2,0);
figure(1)
bode(sys1) % Bode plot from u to x1
figure(2)
bode(sys2) % Bode plot from u to x2

Poles = eig(F_Plant); % The open-loop poles are the eigenvalue of the original plant process matrix
display('The open-loop system transfer function poles are located at:')
display(Poles)
```

```
>> CA1CA3BriefMDemo_SimpDesign
The open-loop system transfer function poles are located at:

Poles =

   -0.4256
   -2.5844

fx >> 
```

Name	Value
BandWidthDesire	3.9190
BesselPoles	[-3.4640 + 2.0000i;-3
ControllabilityMatrix	[0,1;1,-3.0100]
DampRatio	0.8000
F_ClosedLoop	[0,1;-1.2000,-3.1100]
F_Plant	[0,1;-1.1000,-3.0100]
G_Plant	[0;1]
H_Plant_FullState	[1,0;0,1]
H_Plant_State1	[1,0]
H_Plant_State2	[0,1]
ITAEPoles	[-2.8284 + 2.8284i;-2
K_Bessel	[14.8993,3.9180]
K_ITAE	[14.8997,2.6468]
K_LQR	[0.3866,0.2814]
K_SimpleExp	[0.1000,0.1000]
K_SOD	[19.1500,4.1900]
Ks_Bessel	[15.9993,0]
Ks_ITAE	[15.9997,0]
Ks_LQR	[1.4866,0]
Ks_SOD	[20.2500,0]
NatrualFreq	4.5000
Poles	[-0.4514;-2.6586]
Q_LQR	[1,0;0,1]
R_LQR	1
Signal_KX_Bessel	1x2 ss
Signal_KX_ITAE	1x2 ss
Signal_KX_LQR	
Signal_KX_SimpleExp	
Signal_KX_SOD	

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- Current Folder is, by default, where MATLAB save and load data, scripts and functions. It is suggested that a new folder is built for each new project. If a MATLAB script is written for the project, it is suggested to save the script in that folder.

- Start a new MATLAB script by clicking ‘NEW’. Save a script using ‘Ctrl+S’ on the keyboard. Open a script by double click the script file in ‘Current Folder’. Run the active MATLAB script by clicking ‘Run’.

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- Command Window is where you can run MATLAB commands line by line by key in the commands. If errors and warnings happen, the noting messages appear in Command Window. Information about variables in use and programs being executed can be made appear here as well if intended.

- It is equivalent to run a script and to run the commands in the script line by line in the Command Window.

- All the variables defined in the script and Command Window will be temporarily saved in Workspace automatically. MATLAB cannot recognize a variable if it is not in the Workspace before using.

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- The variables in Workspace can be scalar, vector, matrix and other forms. To check the value of a variable, type its name in the Command Window.
- When define a variable, avoid naming it with MATLAB keywords (such as ‘if’, ‘else’, ‘end’) and existing build-in functions (such as ‘roots’, ‘sqrt’).
- MATLAB is case sensitive.

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- Clear the workspace: clear
- Close all the pop-up figures: close all
- Save all the variables in Workspace to the Current Folder:

  save(‘#NAME#.mat’)
- Load variables to Workspace from existing .mat files in Current Folder:

  load(‘#NAME#.mat’)

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- Define a scalar/vector/matrix: (An example)

  #NAME# = 1;   for Scalar

  #NAME# = [1;2;3];   for Column Vector

  #NAME# = [1,2,3];   for Row Vector

  #NAME# = [1,2,3;4,5,6;7,8,9];   for Matrix

Note that if two variables are named identically, the later defined variable will replace the earlier defined variable from the Workspace.

- Make Comments

  #MATLAB CODES#   % #Any Comments#

Whatever is after a ‘%’ in a row would not be compiled and executed, both in scripts and Command Window.

KEY MATLAB SCRIPTS

- SOME BASICS AND FREQUENTLY USED COMMENDS

- Call a MATLAB function:

```
[#Output1#,…,#OutputM#] = #FunctionName#(#Input1#,…,#InputN#);
```

Note that depending on the function, it could have No / Single / Multiple Outputs and Inputs. Even for the same function, it could work in different conditions.

If a function returns several variables and some of them are not used in the later calculation, '~' can be used to replace '#OutputX#' in the output list for those unused variables.

- Check the functionality of a MATLAB function:

help #FunctionName# or doc #FunctionName# (Detailed)

KEY MATLAB SCRIPTS

- OPEN-LOOP ANALYSIS OF THE ORIGINAL PLANT

```
%% ----------------------------------------------------------------------- %
% Open Loop System Analysis

F_Plant = [0,1;-1.10,-3.01]; % Process Matrix
G_Plant = [0;1]; % Input Matrix
H_Plant_State1 = [1,0]; % Output Matrix
H_Plant_State2 = [0,1]; % Output Matrix Assuming x2 is the Output
H_Plant_FullState = [1,0;0,1]; % Output Matrix Assuming Both States are Outputs (for State-Feedback Purpose)

sys1 = ss(F_Plant,G_Plant,H_Plant_State1,0); % Use function 'ss' to establish state-space model
sys2 = ss(F_Plant,G_Plant,H_Plant_State2,0);
figure(1) % Open a new figure window named 'figure 1'
bode(sys1) % Bode plot from u to x1; Use function 'bode' to plot bode graph
figure(2)
bode(sys2) % Bode plot from u to x2

Poles = eig(F_Plant); % The open-loop poles are the eigenvalue of the original plant process matrix
% Use function 'eig' to calculate the eigenvalue of a square matrix
disp('The open-loop system transfer function poles are located at:')
display(Poles)
% Use function 'disp' and 'display' to display messages on Command Window

ControllabilityMatrix = [G_Plant,F_Plant*G_Plant];
if (det(ControllabilityMatrix)==0)
    error('The original plant is not controllable.')
    % An 'error' command will stop the MATLAB program and pop-up an error message
    % If the controllability matrix has determinant of zero, the plant is not controllable.
end
```

Original Plant Analysis

KEY MATLAB SCRIPTS

- SIMPLE STATE-FEEDBACK DESIGN USING ACKERMANN’S FORMULA

```
%% ---------------------------------------------------------------------- %
% Design Using Ackermann's Formula

sys_fullstateplant = ss(F_Plant,G_Plant,H_Plant_FullState,[0;0]);
% The original plant with full state vector as output

K_SimpleExp = [0.1,0.1];
Signal_KX_SimpleExp = ss(0,[0,0],0,K_SimpleExp); % Dummy system representing the value of K*x

sys_fb_SimpleExp = feedback(sys_fullstateplant,Signal_KX_SimpleExp,-1); % Form a feedback system.
% Use function 'feedback' to establish a closed-loop system with a feedback
% controller
% Note that this system is 1-input-2-output system. The input is r (with
% Ks=1 by default) and the outputs are x1 and x2.

figure(3)
bode(sys_fb_SimpleExp) % Bode plot from r to x1 and x2

F_ClosedLoop = F_Plant-G_Plant*K_SimpleExp;
Poles = eig(F_ClosedLoop); % The closed-loop poles are the eigenvalue of the closed-loop process matrix
```

Arbitrarily chosen controller gain [0.1,0.1]

KEY MATLAB SCRIPTS

- SIMPLE STATE-FEEDBACK DESIGN USING ACKERMANN’S FORMULA

```
ITAEPoles = 4*[-0.7071+0.7071*1i;-0.7071-0.7071*1i]; % ITAE Design Poles as a column vector
% In MATLAB, 'a+b*1i' is common way of defining complex value
K_ITAE = acker(F_Plant,G_Plant,ITAEPoles);
% Use function 'acker' to calculate the controller gain using Ackermann's Formula
Signal_KX_ITAE = ss(0,[0,0],0,K_ITAE);
sys_fb_ITAE = feedback(sys_fullstateplant,Signal_KX_ITAE,-1);

Ks_ITAE = 1/dcgain(sys_fb_ITAE); % Calculate the steady state gain of the feedback system
% Use function 'dcgain' to calculate the steady state gain of a system
% The returned value of a function can be directly used as part of
% other calculations.
sys_cl_ITAE = Ks_ITAE*sys_fb_ITAE;
% The feedback system together with the scaling gain Ks forms the final design of closed-loop system
% Note that sys_cl_ITAE is SINGLE output system with output y=x1

figure(4)
bode(sys_cl_ITAE) % Bode plot from r to y
figure(5)
step(sys_cl_ITAE) % Unit step response from r to y; Use function 'step' to observe the step response of a system
grid on % Add grids to the step response figure
```

ITAE Prototype Pole Placement

KEY MATLAB SCRIPTS

- SIMPLE STATE-FEEDBACK DESIGN USING ACKERMANN’S FORMULA

```
% Similarly for Bessel Prototype
BesselPoles = 4*[-0.8660+0.5000*1i;-0.8660-0.5000*1i];
K_Bessel = acker(F_Plant,G_Plant,BesselPoles);
Signal_KX_Bessel = ss(0,[0,0],0,K_Bessel);
sys_fb_Bessel = feedback(sys_fullstateplant,Signal_KX_Bessel,-1);

Ks_Bessel = 1/dcgain(sys_fb_Bessel);
sys_cl_Bessel = Ks_Bessel*sys_fb_Bessel;


figure(6)
bode(sys_cl_Bessel)
figure(7)
step(sys_cl_Bessel)
grid on
```

Bessel Prototype Pole Placement

KEY MATLAB SCRIPTS

- SIMPLE STATE-FEEDBACK DESIGN USING ACKERMANN'S FORMULA

```
DampRatio = 0.8; % Define Damping Ratio for Reference Model
NatrualFreq = 4.5; % Define Natural Frequency for Reference Model
BandWidthDesire = NatrualFreq*sqrt((1-2*DampRatio^2)+sqrt((1-2*DampRatio^2)^2+1));
% Calculate the Bandwidth of the reference model
SODPoles = roots([1,2*DampRatio*NatrualFreq,NatrualFreq^2]);
% Calculate the pole positions of the reference model, which is the root of
% pole polynomial
% Use function 'roots' to calculate the roots of a polynomial
K_SOD = acker(F_Plant,G_Plant,SODPoles);
Signal_KX_SOD = ss(0,[0,0],0,K_SOD);
sys_fb_SOD = feedback(sys_fullstateplant,Signal_KX_SOD,-1);

Ks_SOD = 1/dcgain(sys_fb_SOD);
sys_cl_SOD = Ks_SOD*sys_fb_SOD;

figure(8)
bode(sys_cl_SOD)
figure(9)
step(sys_cl_SOD)
grid on
```

Second-Order Reference Model Based Pole Placement

KEY MATLAB SCRIPTS

- SIMPLE STATE-FEEDBACK DESIGN USING LQR

```
%% ---------------------------------------------------------------------- %
% Design Using LQR

Q_LQR = [1,0;0,1]; % Define penalty matrix Q
R_LQR = 1; % Define penalty matrix R (in this case, a scalar)
[K_LQR,~,~] = lqr(F_Plant,G_Plant,Q_LQR,R_LQR);
% Use function 'lqr' to calculate the controller gain for LQR approach
% 'lqr' is a multiple output function and only second/third output is not
% used in the rest of the script
Signal_KX_LQR = ss(0,[0,0],0,K_LQR);
sys_fb_LQR = feedback(sys_fullstateplant,Signal_KX_LQR,-1);

Ks_LQR = 1/dcgain(sys_fb_LQR);
sys_cl_LQR = Ks_LQR*sys_fb_LQR;

figure(10)
bode(sys_cl_LQR)
figure(11)
step(sys_cl_LQR)
grid on
```

LQR Controller Design

KEY MATLAB SCRIPTS

- STATE-FEEDBACK DESIGN INTRODUCING STATE AUGMENTATION

```
%% ---------------------------------------------------------------------- %
% Formulate the Augmented State-Space Model

Fbar = [0,1,0;-1.10,-3.01,0;1,0,0]; % The process matrix for the augmented system
Gu = [0;1;0]; % The input matrix for u
Gr = [0;0;-1]; % The input matrix for r
Gv = [0;1;0]; % Used for Disturbance Analysis; The input matrix for disturbance v
Hbar = [1,0,0]; % The output matrix for y
```

Establishing the Augmented State Space Model

KEY MATLAB SCRIPTS

- STATE-FEEDBACK DESIGN INTRODUCING STATE AUGMENTATION

```
%% ---------------------------------------------------------------------- %
% Design Using Ackermann's Formula

ITAEPoles = 3*[-0.7081;-0.5210+1.068*1i;-0.5210-1.068*1i];
K_ITAE = acker(Fbar,Gu,ITAEPoles);
sys_ag_ITAE = ss(Fbar-Gu*K_ITAE,Gr,Hbar,0);
% sys_ag_ITAE is already closed-loop system since 'F-GK' is used as the
% process matrix
% There is no need to use function 'feedback' to build the closed-loop
% system
figure(12)
bode(sys_ag_ITAE)
figure(13)
step(sys_ag_ITAE)
grid on

BesselPoles = 3*[-0.9420;-0.7455+0.7112*1i;-0.7455-0.7112*1i];
K_Bessel = acker(Fbar,Gu,BesselPoles);
sys_ag_Bessel = ss(Fbar-Gu*K_Bessel,Gr,Hbar,0);
figure(14)
bode(sys_ag_Bessel)
figure(15)
step(sys_ag_Bessel)
grid on
```

ITAE and Bessel Prototype Pole Placement

KEY MATLAB SCRIPTS

- STATE-FEEDBACK DESIGN INTRODUCING STATE AUGMENTATION

```
SODPoles = [-12;-3.6+2.7*1i;-3.6-2.7*1i];
K_SOD = acker(Fbar,Gu,SODPoles);
sys_ag_SOD = ss(Fbar-Gu*K_SOD,Gr,Hbar,0);
figure(16)
bode(sys_ag_SOD)
figure(17)
step(sys_ag_SOD)
grid on
```

Second-Order Dominant Pole Approach

KEY MATLAB SCRIPTS

- STATE-FEEDBACK DESIGN INTRODUCING STATE AUGMENTATION

```
%% ---------------------------------------------------------------------- %
% Design Using LQR

Q_LQR = [1,0,0;0,1,0;0,0,1];
R_LQR = 1;
[K_LQR,~,~] = lqr(Fbar,Gu,Q_LQR,R_LQR);

sys_ag_LQR = ss(Fbar-Gu*K_LQR,Gr,Hbar,0);
figure(18)
bode(sys_ag_LQR)
figure(19)
step(sys_ag_LQR)
grid on

sys_ag_LQR_Control = ss(Fbar-Gu*K_LQR,Gr,-K_LQR,0);
% In sys_ag_LQR_Control, Hbar is changed to -K_LQR
% In this case, MATLAB would treat -K_LQR*x as the output of the system,
% which is nothing else but the control signal
% Therefore, by doing a unit step response analysis, the control signal
% response can be displayed
figure(20)
step(sys_ag_LQR_Control)
grid on
```

LQR Controller Design for Augmented System

KEY MATLAB SCRIPTS

- STATE-FEEDBACK DESIGN INTRODUCING STATE AUGMENTATION

```
%% ---------------------------------------------------------------------- %
% Disturbance Study

sys_ag_Disturb = ss(Fbar-Gu*K_LQR,Gv,Hbar,0);
% Gr is changed to Gv
% For sys_ag_Disturb, the input matrix is set Gv in order to study the
% system response when disturbance v comes in
% In this case study, the reference signal r is zero.
figure(21)
bode(sys_ag_Disturb)
figure(22)
step(sys_ag_Disturb)
grid on
sys_ag_Disturb_Control = ss(Fbar-Gu*K_LQR,Gv,-K_LQR,0);
figure(23)
step(sys_ag_Disturb_Control)
grid on
```

Disturbance Analysis for the LQR Controller based on Augmented System

REQUIREMENTS AND QUESTIONS FOR THE REPORT

- Note that this brief note serves only as an instruction. **The note itself is seriously lack of case studies, analysis and explanation to all the phenomenon observed in the simulation**.

- In the report, **individual observation, thinking, explanation, analysis and conclusion are expected**.

- ***CA1 report and CA3 report need to be submitted SEPARATELY via IVLE.***

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA1)

- For the original plant, use the MATLAB function BODE to obtain plots of the frequency response from u to x_1 and from u to x_2.
- Use $u = -Kx, K = [0.1 \quad 0.1]$ as the control signal and use function FEEDBACK to obtain the closed-loop system. Plot the frequency response of the closed-loop system.
- Based on the closed-loop specifications required, choose your desired closed-loop pole locations using the ITAE criterion. Use Ackermann’s formula to calculate the state-feedback gains to place the poles of the closed-loop at the desired locations.
- Provide a listing of how you used MATLAB to do the computations in the Ackermann’s formula and the results of the computations.

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA1)

- Provide calculations to show how the scaling gain was chosen.
- Provide frequency response plots of the final closed-loop. You may have to go through several iterations of choice of desired pole locations to achieve the specifications. If so, provide representative plots.
- In addition to the ITAE methodology, experiment further with Bessel prototype table methodology and Second-Order Dominant Response methodology. Document, describe and discuss all your experimentations !!

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA1)

- In LQR design, choose a set of weighting gains (penalties) for the LQR problem.
- Use MATLAB function LQR to compute the state-feedback gains that will minimize the cost function.
- Form the feedback loop for this set of gains and check if it satisfies the closed-loop specifications.
- If the specs are not met, choose another set of weightings.
- Finally, choose the scaling gain to meet the steady-state condition.
- Provide the plots and listings that are representative of your efforts in this section. You may restrict your design to only using a diagonal matrix Q in the LQR formulation. Document, describe and discuss all your experimentations.

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA3)

- Use the method of state-augmentation to design a controller for the plant. Explain carefully why state augmentation will result in a closed-loop system meeting the specs for the two items of steady-state requirement. (Then, note that since the system is linear, it suffices to consider disturbance as zero to meet the remaining frequency response requirements.)
- First, use ITAE criterion to select the closed loop poles and then use Ackermann’s formula to calculate the state feedback gains.
- Second, use LQR design method (with a diagonal matrix for Q). Provide a listing (with sufficient comments) on how the MATLAB package was used in the design for this part. Investigate the effect of varying each of the weighting parameters on the frequency response between r and y.

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA3)

- For both methods, provide frequency response plots showing the frequency response between r and y. You should at least have one set of resulting controller gains that meet the required specs.
- Provide plots showing the response of the output, y, to a unit step in r.
- Provide plots showing the response of the output, y, to a unit step in the unmeasurable disturbance, v. For this case, additionally provide a plot of the control input, u, in response to the step in the unmeasurable disturbance.
- In all cases, as far as possible, provide listing to show how you used MATLAB to generate various variables and plots.

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA3)

- In addition to the ITAE and LQR methodologies, experiment further with Bessel prototype table methodology and Second-Order dominant response methodology. Document, describe and discuss all your experimentations.
- Why does it make sense to require a lower closed-loop bandwidth (1.5 rad/s) here? Experiment, explore and discuss.
- Calculate the closed-loop transfer functions attained in CA3 and discuss all pertinent aspects.

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA3)

- In all of the above, note that your design were based on:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1.10 & -3.01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Consider next, that unknown to you, the system has in fact changed to:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1.23 & -3.51 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1.10 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

(see next page)

REQUIREMENTS AND QUESTIONS FOR THE REPORT (CA3)

Explore, discuss, analyze with all suitable simulations, analysis and descriptions in this type of situation on the use of the methods of:

- using a ‘scaling gain’
- using the augmented state variable $\dot{x}_I = y - r$

Provide a suitable comparative exploration/discussion.

APPENDIX: ITAE AND BESSEL PROTOTYPE POLE LOCATION

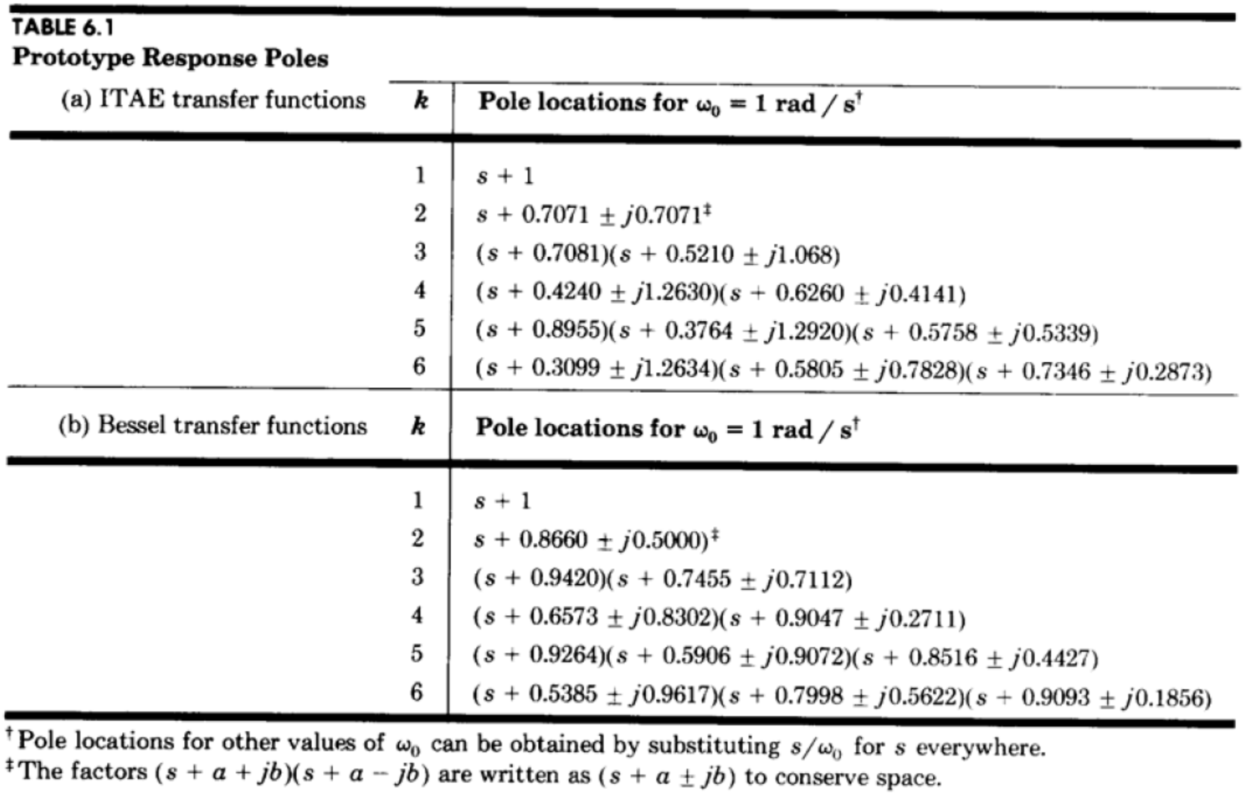


TABLE 6.1

Prototype Response Poles

(a) ITAE transfer functions	k	Pole locations for $\omega_0 = 1$ rad / s[†]
	1	$s + 1$
	2	$s + 0.7071 \pm j0.7071$[‡]
	3	$(s + 0.7081)(s + 0.5210 \pm j1.068)$
	4	$(s + 0.4240 \pm j1.2630)(s + 0.6260 \pm j0.4141)$
	5	$(s + 0.8955)(s + 0.3764 \pm j1.2920)(s + 0.5758 \pm j0.5339)$
	6	$(s + 0.3099 \pm j1.2634)(s + 0.5805 \pm j0.7828)(s + 0.7346 \pm j0.2873)$

(b) Bessel transfer functions	k	Pole locations for $\omega_0 = 1$ rad / s[†]
	1	$s + 1$
	2	$s + 0.8660 \pm j0.5000)$[‡]
	3	$(s + 0.9420)(s + 0.7455 \pm j0.7112)$
	4	$(s + 0.6573 \pm j0.8302)(s + 0.9047 \pm j0.2711)$
	5	$(s + 0.9264)(s + 0.5906 \pm j0.9072)(s + 0.8516 \pm j0.4427)$
	6	$(s + 0.5385 \pm j0.9617)(s + 0.7998 \pm j0.5622)(s + 0.9093 \pm j0.1856)$

[†] Pole locations for other values of ω_0 can be obtained by substituting s/ω_0 for s everywhere.

[‡] The factors $(s + a + jb)(s + a - jb)$ are written as $(s + a \pm jb)$ to conserve space.

KEY REFERENCES

- Lecture Notes of EE4302 Advanced Control Systems
- CA1, CA3 Experiment Requirement Sheet for Special Semester May/June 2016
- EE4302 Advanced Control Systems Experiment 1 Briefing Notes by Dr. K.Z. Tang & Prof. T.H. Lee

THANK YOU